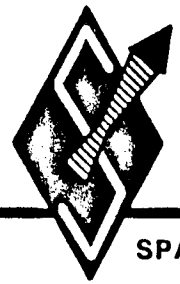
SPARTA, INC.

TRANSPUTER FINITE ELEMENT SOLVER

SPARTA, Inc.
4901 Corporate Drive
Huntsville, AL 35805
(205) 837-5200

1187-005/C



SPARTA, INC.

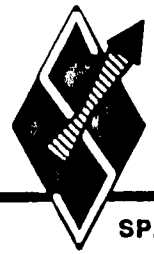
INTRODUCTION

- SPARTA SBIR AWARD
- GENERAL TRANSPUTER INFORMATION
- RESULTS OF FEASIBILITY STUDY
- PROSPECTS FOR A LARGE-SCALE TFES

Introduction--Transputer Based Finite Element Solver

In January 1987, SPARTA received a Phase I SBIR award from the NASA Lewis Research Center to investigate the feasibility of a finite element solver implemented on multiple VLSI processors. The transputer was chosen as the processor for the feasibility study since it combined low cost with high performance and was specifically designed to directly link with other transputers to form networks of multiple processors. This presentation consists of three parts:

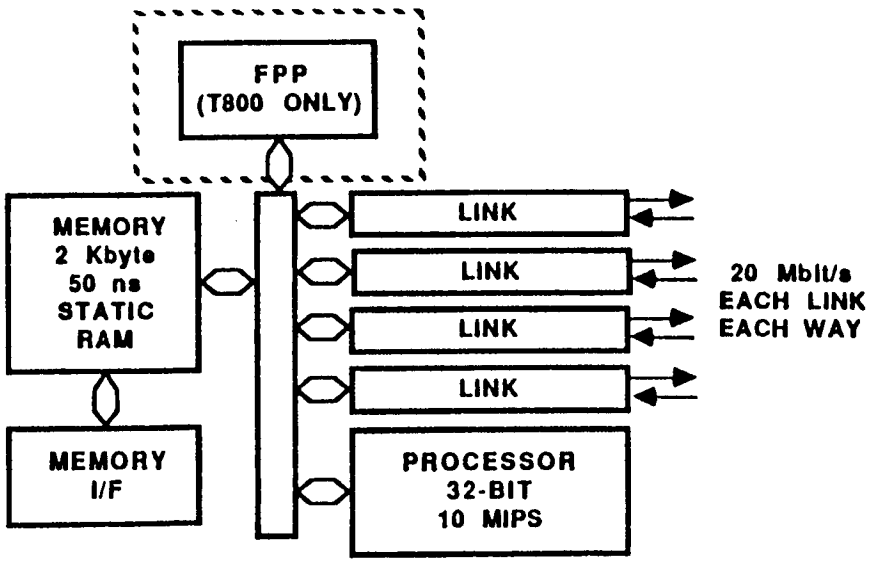
(1) a brief description of transputers,
(2) a summary of the SBIR feasibility study, and
(3) a discussion of issues concerning a large scale transputer based finite element solver (TBFES) and the performance levels which can be expected.



SPARTA, INC.

GENERAL INFORMATION - INMOS T414 AND T800 TRANSPUTERS

- T414/T800 SPECIFICATIONS
  - 10 MIPS
  - 4 BIDIRECTIONAL LINKS @ 20 Mbits/sec
  - 4 Gbyte ADDRESS SPACE
  - 1.2 MICRON CIRCUITRY
  - TRANSPUTER CHIP OCCUPIES ONLY 2 SQUARE INCHES
- T800 FPP CAN SUSTAIN 1.5 Mflops
- PROGRAMMING LANGUAGES
  - OCCAM
  - PASCAL
  - C
  - FORTRAN
  - FORTH



TRANSPUTER CHIP CONFIGURATION

Transputer General Information

Specifications

The transputer, developed by INMOS Corporation, is a complete VLSI computer on a single chip. Two 32-bit versions exist: the T414, which has an integer processor, and the T800, which has a floating point processor. Transputers have 10 MIP processors, 2 (T414) or 4 (T800) Kbytes of static cache memory, interfaces to external RAM through a 32 bit address bus (to access up to 4 Gbytes of memory), and four bidirectional serial ports, called links, to directly communicate with other transputers. Links are capable of transmitting data a rate of 20 Mbits/sec. Its 1.2 micron circuitry enables a transputer chip to occupy less than two square inches on a printed circuit board; four transputers with 1 Mbyte of RAM each can fit on a single IBM PC plug-in card. The T800 is capable of sustaining 1.5 million floating point operations per second (Mflops), so a plug-in card with four T800's has 6 Mflops--about 18 times the sustained floating point performance of a VAX 11/780.

Languages

The transputer architecture was designed to implement occam, a high-level structured language for parallel processing. Occam is strongly typed and has a number of built-in constructs useful for defining parallel tasks, for communicating between parallel tasks (on the same transputer or on other transputers connected to it through links), and for real-time control of program execution. Since occam maps closely to the transputer architecture, programs written in occam are very efficient and take full advantage of the transputer's unique capabilities. A few desirable features have not yet been implemented, however. The current release of occam (occam2) does not have structured variables or pointers, does not allow recursion, and does not support functions. In addition, the lack of a transputer operating system and imperfect error locators makes debugging very challenging.

Pascal, C, FORTRAN and Forth compilers are also available.

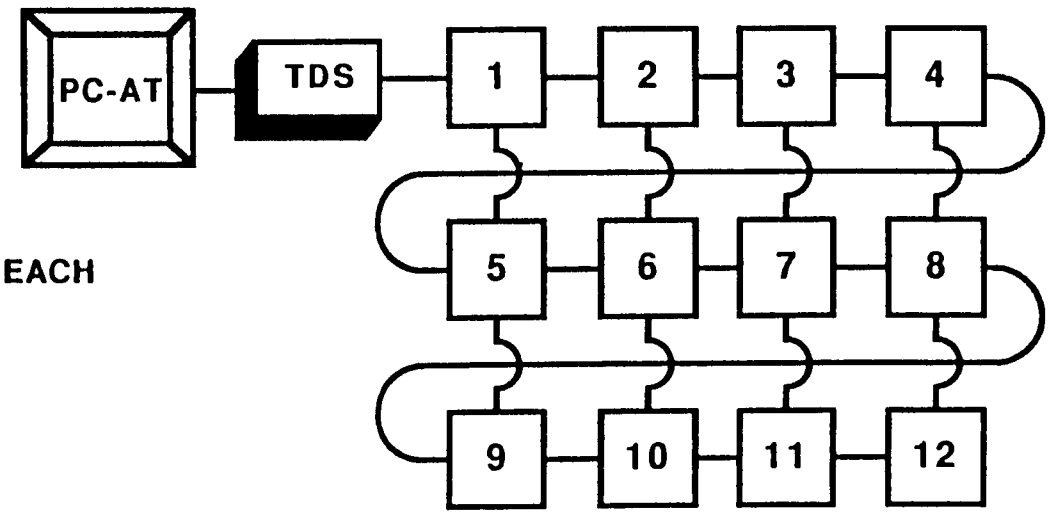
HARDWARE FOR SBIR FEASIBILITY STUDY

SPARTA, INC.

HARDWARE:

- IBM PC-AT CLONE
- ONE T414A TDS WITH 2 MBytes
- TWELVE T414Bs WITH 256 KBytes EACH
- TOTAL SYSTEM COST OF $15,000

A TRANSPUTER BASED FINITE ELEMENT SOLVER



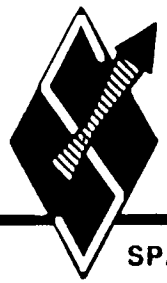
TRANSPUTER NETWORK
ARRANGED IN A HELICAL GRID

Hardware

The hardware used for the feasibility study included an IBM PC-AT compatible computer, an INMOS Transputer Development System (TDS) board with a T414 transputer and 2 Mbytes of RAM, and twelve T414's with 256 Kbytes of RAM each. The AT compatible served as the host computer by providing keyboard, screen and disk access to and from the TDS; the PC microprocessor and RAM memory are not used for any transputer operation. The thirteen transputers were configured in a helical grid, a topology which has relatively short communication path lengths.

Towards the end of the study, SPARTA received a pre-production prototype T800 for testing and evaluation of its floating point processor. This T800 was a revision "A" and ran at only 0.6 Mflops, less than half of the current production T800 (revision "C") capability. Sequential versions of the parallel finite element code (derived by eliminating task distribution) was run on both the T800A and the TDS's T414A transputers for comparison with the parallel code running on the entire network of T414's.

The minicomputers used for comparison were an Apollo DN660 and a VAX 11/780 with a floating point accelerator (FPA). The finite element code on these machines, although written in FORTRAN, is a direct translation of the occam code running on the transputer FE solvers (neglecting communication and parallel logic code) and executes the same sequence of steps to arrive at a solution. Timing results were obtained when both minicomputers had no other loads so the reported CPU times were equal to the actual elapsed times.

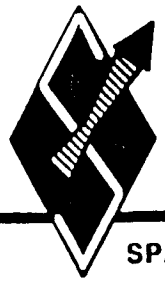


SPARTA, INC.

SBIR FEASIBILITY DEMONSTRATION

- GOALS
  - INVESTIGATE WAYS OF DECOMPOSING FE METHOD ON A NETWORK OF TRANSPUTERS
  - IMPLEMENT A TBFES
  - COMPARE COST AND PERFORMANCE TO EXISTING COMPUTER SYSTEMS

THIS PAGE LEFT BLANK INTENTIONALLY



SPARTA, INC.

FE SOLUTION METHODS

- DIRECT METHOD
  - ASSEMBLY OF GLOBAL STIFFNESS MATRIX
    - -- HIGHLY PARALLEL
    - -- COMPUTE BOUND
  - SOLUTION OF LINEAR EQUATIONS WITH GAUSSIAN ELIMINATION
    - -- GAUSS-JORDAN ELIMINATION HAS EFFICIENT PARALLEL IMPLEMENTATION
    - -- COMPUTE BOUND
- FRONTAL METHOD
  - LOWER OPERATIONS COUNT
  - MULTIPLE CONCURRENT FRONTS

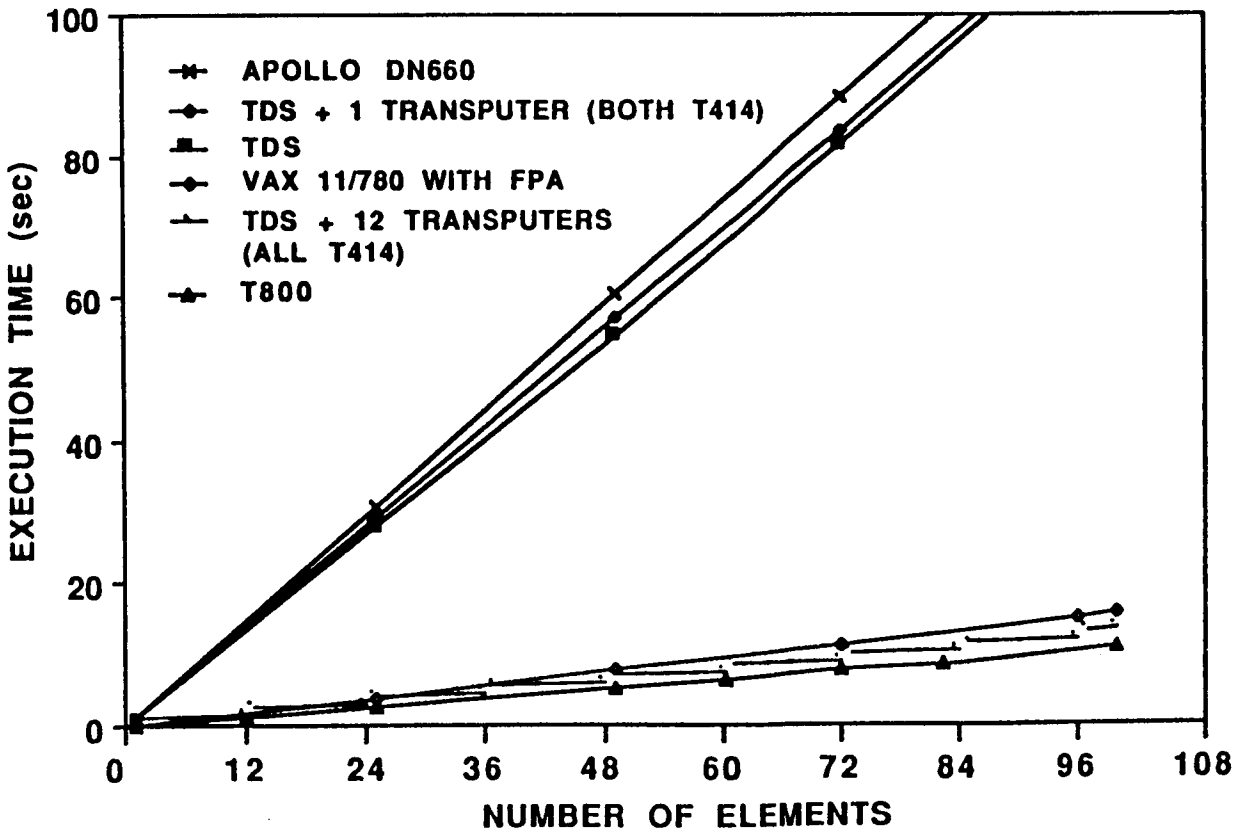
Solution Methods

Two methods of finding solutions to finite element equations showed potential for parallel implementation. The standard method of assembling a global stiffness matrix and solving the whole system of equations with a Gaussian elimination-based equation solver showed inherent parallelism in both global matrix assembly and matrix solution parts. The Frontal Method was also considered since it appeared that multiple solution fronts could be initiated concurrently.

A complete FE solver based on assembling a global stiffness matrix and solving the structural equations with both Gauss-Jordan elimination and inversion was implemented on the 13 transputers and compared with functionally identical programs running on the VAX and Apollo computers. Its design and performance are described on the next three viewgraphs. No results are given for the frontal solver since an effective parallel implementation scheme was not resolved during the SBIR study.

EXECUTION TIMES FOR GLOBAL STIFFNESS MATRIX ASSEMBLY

SPARTA, INC.



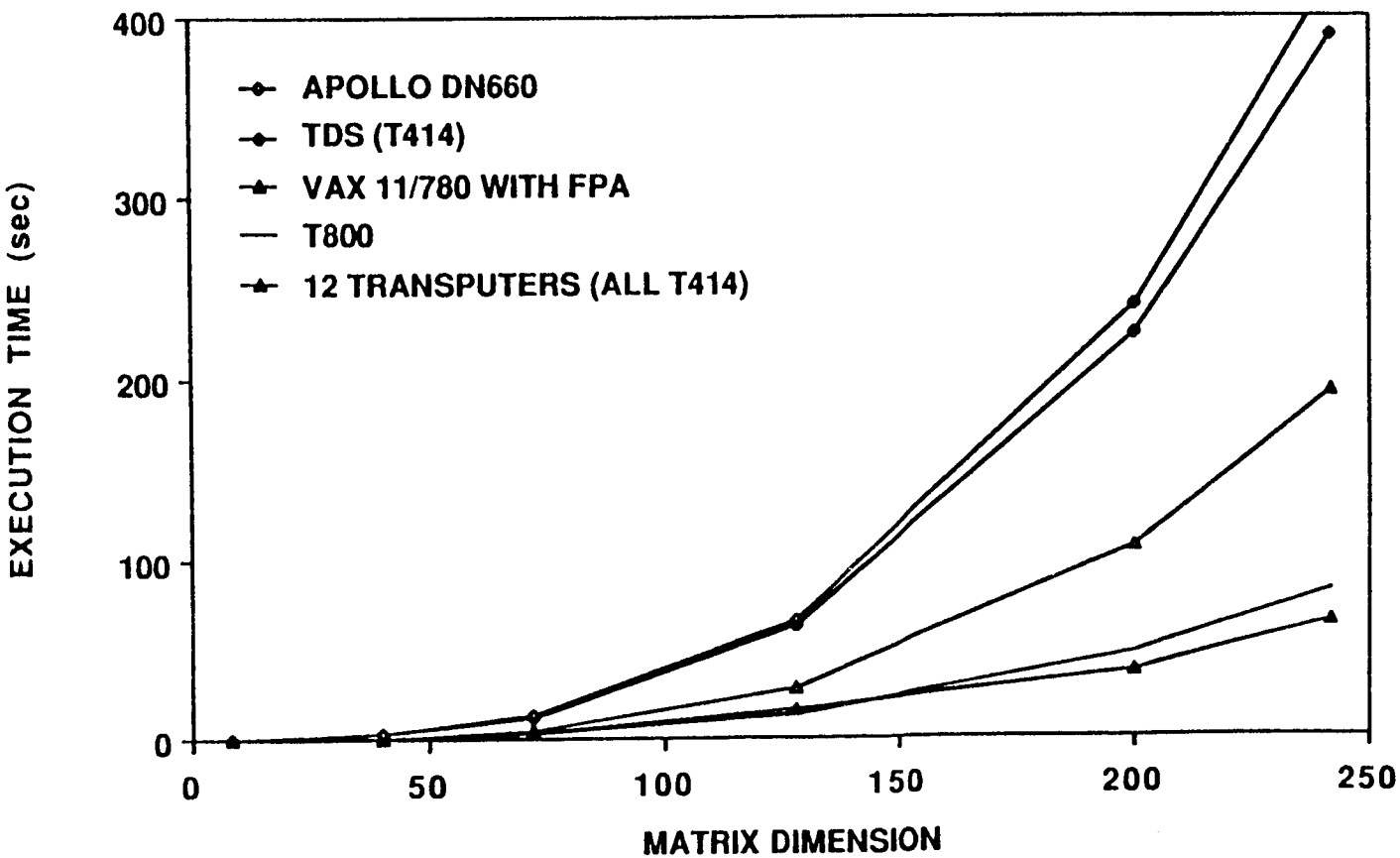
Parallel Assembly of the Global Stiffness Matrix

The assembly of a global stiffness matrix consists of two parts: the calculation of element stiffness matrices (2-D, four node isoparametric elements were used in the SBIR study) and the summation of corresponding terms of the element matrices into one global matrix. The first part, the calculation of element matrices, is completely parallel since element stiffness matrix calculations are entirely idependent of each other and can be done concurrently. The second part, the summation of corresponding terms, requires interprocessor communication but can also be written to be highly parallel. The first part however, is at least 1000 times more compute intensive than the second part, so even if only element stiffness calculations were distributed and summation were performed sequentially, a near 100% efficient parallel implementation could be obtained. Such an approach was taken in the TBFES: the TDS sends 12 elements to the network at a time (one element per processor), lets the network calculate element stiffness matrices, waits for the network to send matrices back, and immediately sends out another set of 12 elements. While the network is busy calculating, the TDS sums the element matrices of the previous set into a global matrix.

Results are given for the Apollo, VAX, T800A, the TDS running sequential code, the TDS with a network of one T414 running parallel code, and the TDS with a network of 12 T414's running parallel code. The results show that, for 100 elements, a single T414 transputer runs faster than the Apollo, and a single T800 (running at only half the speed of production versions) is faster than the VAX. The step pattern in the 12 transputer curve is indicative of the work assignments in the parallel code. The network takes roughly the same amount of time to calculate 12 or fewer elements so the step slope is close to horizontal; a finite jump in execution time occurs after a multiple of twelve is exceeded, however, since after that point some transputers have more elements for which to calculate matrices than do others.

EXECUTION TIMES FOR GAUSS-JORDAN MATRIX INVERSION

SPARTA, INC.



Parallel Solution of Linear Equations

Gauss-Jordan elimination and Gauss-Jordan inversion, both with partial pivoting, were implemented on the transputer network. No attempts were made at taking advantage of the matrices' sparse, symmetric nature, so the implemented algorithms can be used to solve any well-conditioned set of linear equations.

Gauss-Jordan elimination and inversion can both be implemented on parallel processors with great efficiency since they can be easily load balanced, and since they are compute bound if the number of equations is much larger than the number of processors. The matrix is first divided up among the processors in the network. The matrix in the TBFES was subdivided by distributing entire matrix columns to the transputers, although several ohter schemes are also possible (row distribution, block distribution). Columns of the matrix must be distributed so that the work load is balanced. Elimination is slightly more complicated to load balance than inversion, since elimimation allows columns of the matrix to become inactive as the solution progresses from left to right. If some processors have only inactive columns, then they will become idle and lower the overall efficiency of the program. Columns in the TBFES were distributed in an alternating sequence so that at any stage during elimination, all processors will have equal numbers of active and inactive columns (plus or minus one column). For example, if 4 processors had to solve a system of 26 equations, the column distibution scheme would result in the following:

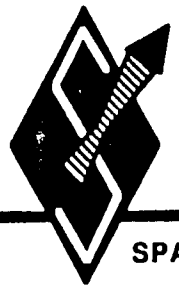
Processor	Columns
1	1, 5, 9, 13, 17, 21, 25
2	2, 6, 10, 14, 18, 22, 26, 27
3	3, 7, 11, 15, 19, 23
4	4, 8, 12, 16, 20, 24

The 27th column is the load vector and resides on the same node as the last coefficient column.

Solution begins after all columns are distributed. The first node then becomes the "pivoting" node and all others become "adjusting" nodes. The pivot node eliminates terms of the first column (which contains the first pivot) and sends the necessary multipliers to the adjusting nodes so they can perform the same operations on all their columns. After zeroing the first column, the pivot node becomes an adjusting node and tells the node containing column 2 to assume the role of the pivot node by eliminating column 2. This process continues until the last coefficient column is eliminated.

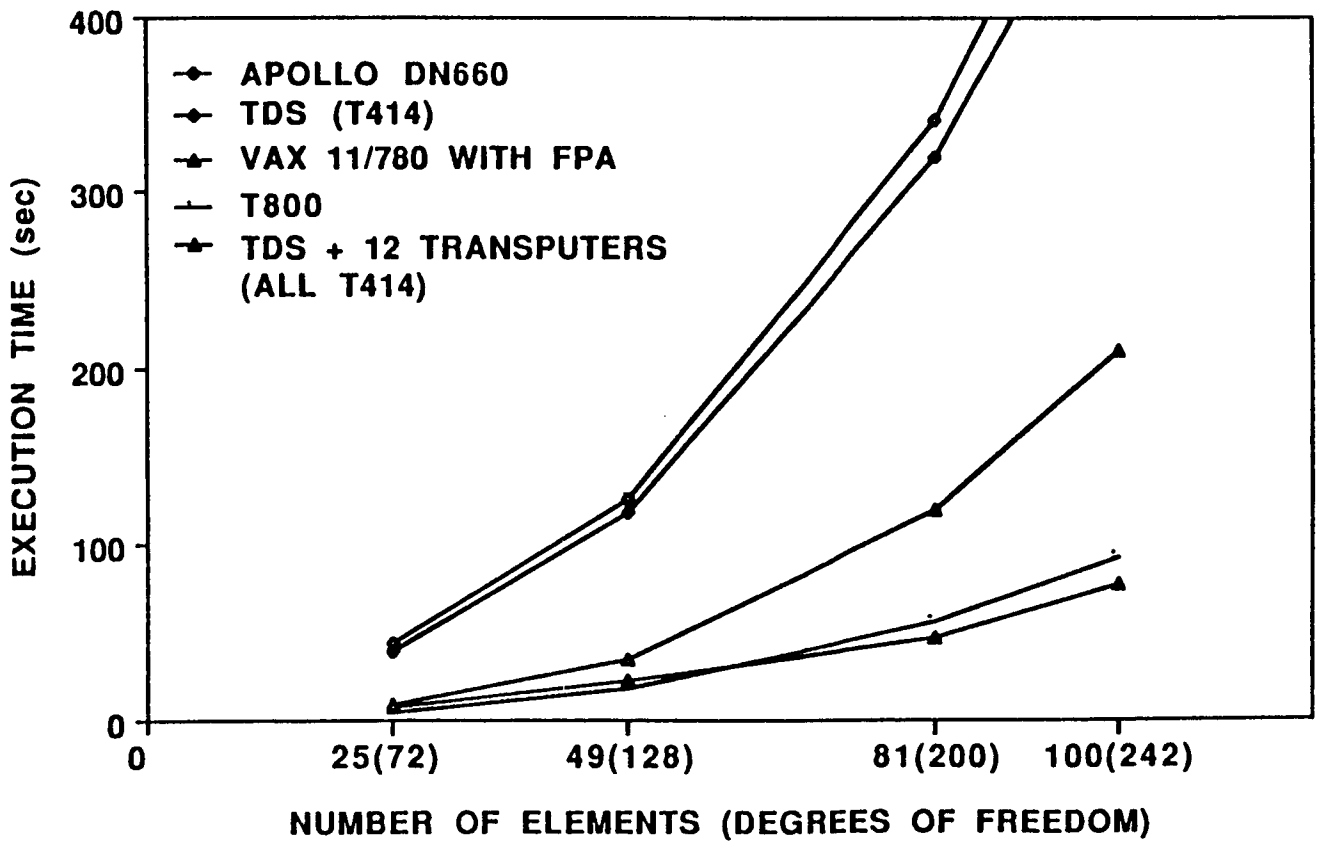
Load balancing Gauss-Jordan inversion was trivial because it was written to keep the entire matrix continuously active--inversion was done "in place" so that the inactive columns of the original matrix were directly replaced with the active columns of the inverse.

Efficiency of parallel equation solvers increases as the number of columns per processor increases, because the computational effort increases with the cube of the number of equations (= number of columns) while communication increases with the product of the number of processors and the number of equations. For extremely large systems of equations solved on a small network of parallel processors, communication is negligible compared to the enormous computational effort and efficiency should reach 100%.



SPARTA, INC.

EXECUTION TIMES FOR COMPLETE FE PROBLEM

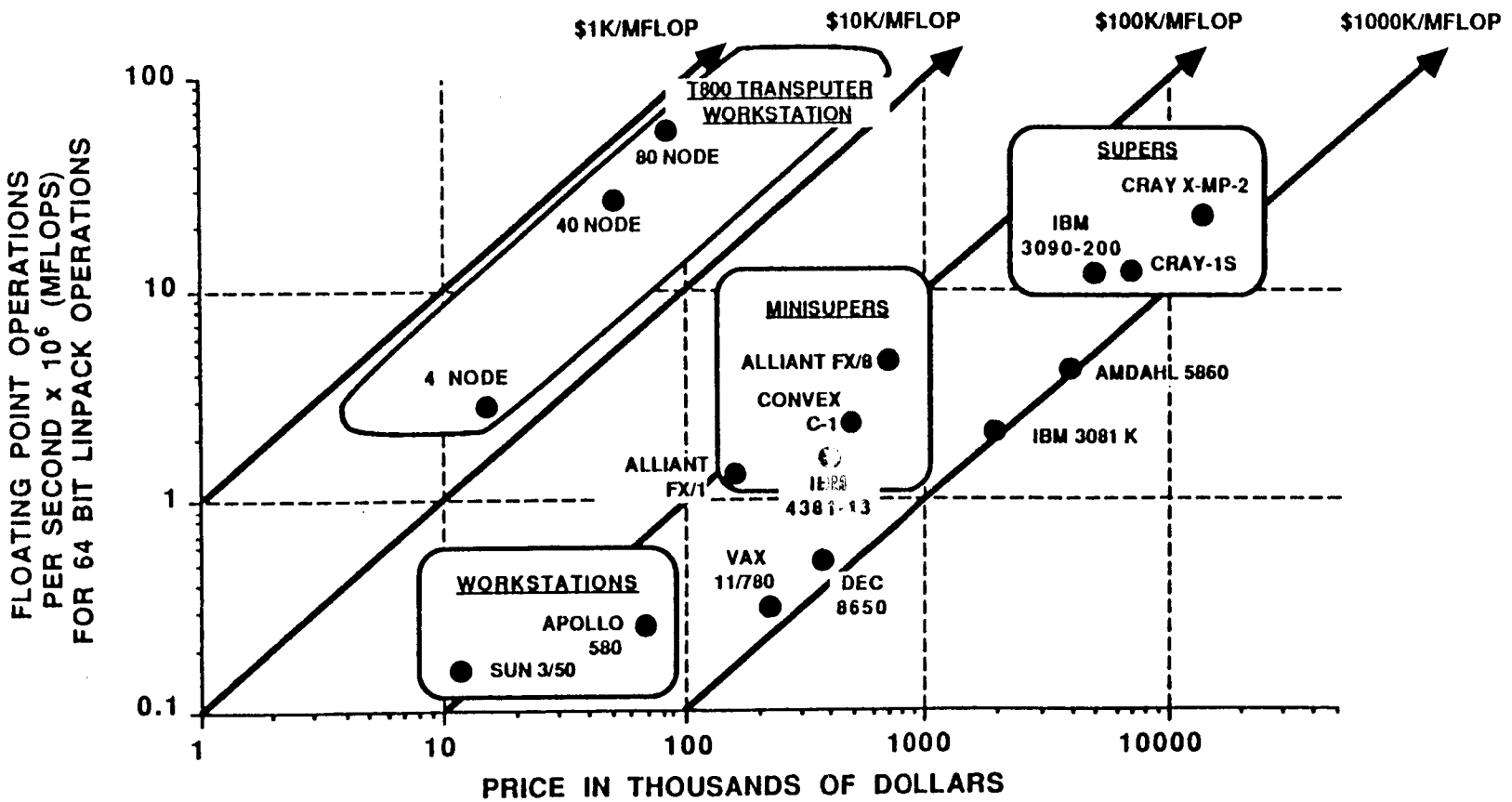


Complete Finite Element Solution

Times for the solution of a complete FE problem are obtained by adding the time it takes to assemble the global stiffness matrix and the time it takes to solve the linear equations. The graph shows this superposition of times using the matrix inversion solution method. The transputer network solves 242 degree of freedom problems 2.7 times faster than the VAX and more than 11 times faster than the Apollo.

PERFORMANCE VERSUS PRICE FOR SEVERAL COMPUTING SYSTEMS

SPARTA, INC.



A Large-Scale, Practical Transputer FE Solver

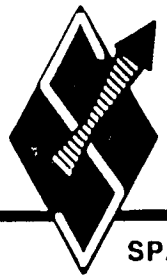
It is evident that transputers have enourmous potential; networks of T800's can be assembled at a rate of $1000 per Mflop, making supercomputer performance levels available at minicomputer prices. Finite element analysis is in a unique position to fully exploit the advantages offered by a large transputer network: the FE Method is compute-intensive and has inherent parallelism; and there is such a pressing need for more powerful finite element solving capabilities that an inexpensive, super-powerful FE solver would be in great demand in the scientific and engineering communities. This section addresses some issues critical to the implementation of an effective FE solver on a large network of transputers.

SPARTA, INC.

VARIABLES AFFECTING PERFORMANCE OF PARALLEL FE SOLVERS

- PROCESSOR SPECIFICATIONS
  - MFLOPS AVAILABLE
  - NUMBER OF LINKS
  - COST OF PROCESSOR
  - COST ON IN-CORE MEMORY FOR EACH PROCESSOR
- INPUT/OUTPUT SPECIFICATIONS
  - LINK TRANSFER RATE
  - MASS STORAGE TRANSFER RATE
  - COST OF MASS STORAGE
- NETWORK GRANULARITY
- SOLUTION METHOD
  - DIRECT FACTORIZATION METHODS
  - FRONTAL METHOD
  - ITERATIVE METHODS

Many variables would influence the performance of a large scale parallel FE solver. These variables, which fall in four general categories (processor specifications, network granularity, input/output specifications, and the solution method used), must be balanced against each other to yield the optimum parallel solver. Finding the appropriate combination is difficult, especially since the slightest improvement in the implementation of a solution method may require a considerable hardware modification. The parameter balancing problem will most likely be iterative, but a feasible initial approach might be to allocate available funds so that a low granularity network (few processors, much RAM available to each processor) with enough memory to solve the largest problems is assembled. Two options are possible if the required amount of in-core memory is unobtainable for a low-granularity network: external mass storage could be added to the processors (maintaining low granularity and using an out-of-core solver), or the number of processors could be increased until enough RAM is accumulated (allowing granularity to increase and using an in-core solver). The final hardware configuration is then used as a basis for selecting, modifying and/or designing a solution method.



SPARTA, INC.

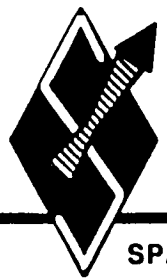
LARGE SCALE FE SOLVER REQUIRES MUCH MEMORY

TWO APPROACHES TO SATISFYING MEMORY REQUIREMENTS

- IN-CORE:
  - ADD RAM TO EACH PROCESSOR
  - ADD PROCESSORS UNTIL ENOUGH RAM IS ACCUMULATED
- OUT-OF-CORE: ADD MASS STORAGE DEVICES TO THE PROCESSOR NETWORK

Memory Requirements

Large-scale FE analysis requires extraordinary amounts of memory. Although transputer networks can be made so large that the collective in-core memory is sufficient to store and solve the entire problem (a $1,000,000 transputer network--1000 T800's with 1 Mbyte each--has 1 Gbyte of in-core memory, sufficient for a 100,000 degree of freedom double-precision problem stored with a half bandwidth of 1250), such high granularity networks require proportionately higher amounts of communication making high efficiency less obtainable. Mass storage devices added to relatively smaller networks of transputers create problems of their own. The slow throughput rates typical of external storage could starve the transputer network of data, making the network as slow as a sequential processor. The throughput rate could be increased by adding more external storage devices, but this quicky raises the cost of a parallel FE solver. If throughput rates have to be so high that each processor requires its own storage device (as might be necessary for an iterative solution method), the price of such a network would keep its size, and therefore its power, low.



EXISTING SOLUTION METHODS HAVE DRAWBACKS

SPARTA, INC.

- DIRECT METHODS
  - GAUSS-JORDAN ELIMINATION
    - -- EFFICIENT PARALLEL IMPLEMENTATION
    - -- INEFFICIENT FOR FE ANALYSIS
  - LUD DECOMPOSITION
    - -- BAND METHOD HAS EFFICIENT PARALLEL IMPLEMENTATION IF BW >> # Xp's
      - - DOES NOT TAKE ADVANTAGE OF SPARSENESS WITHIN THE BAND
      - - DOES NOT TAKE ADVANTAGE OF LOCAL VARIATION IN BANDWIDTH
    - -- ENVELOPE AND BLOCK METHODS ARE MORE EFFICIENT, BUT
      - - INCREASE IN EFFICIENCY IS PROBLEM DEPENDENT
      - - LOAD BALANCING BECOMES PROBLEM DEPENDENT
  - FRONTAL METHOD
    - -- LOW OPERATIONS COUNT
    - -- VARIABLE FRONTWIDTH MAKES LOAD BALANCING DIFFICULT
    - -- ADDITIONAL BOOKKEEPING MAY BE PROHIBITIVE

Selecting a Solution Method--Load balancing, Communications and Efficiency

Tapping the full potential of a large transputer network is possible only if (1) each processor contributes to the solution of the entire problem and is never idle, and (2) the processors spend nearly all their time performing useful calculations rather than transmitting information to other processors. These two criteria which determine the overall efficiency of parallel solvers are often difficult to satisfy simultaneously. Although one part of the FE Method, the generation of element stiffness matrices, is so parallel that it can easily satisfy the conditions governing efficiency, the more time-consuming part of solving the structural equations does not readily yield to efficient implementation. A few of the advantages and disadvantages of several solution methods to structural equations (having sparse, symmetric, banded, positive-definite coefficient matrices) are given below:

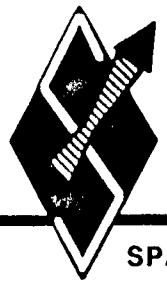
Gaussian Elimination: High efficiency only if the bandwidth is much larger than the number of processors. A new set of constraints requires the entire problem to be solved again.

Gauss-Jordan Elimination: Easily load balanced, but has a high operations count. A new set of constraints requires the entire problem to be solved again.

LU or Cholesky Decomposition: Factorization methods are preferable to straightforward elimination since new constraints can be solved for in a quick back-substitution step. A few of its variations are described below:

- Band Method: Efficient only if the bandwidth is much larger than the number of processors. Since the band is always of a known geometry, it is easy to divide up among processors for load balancing. This method does not take advantage of sparseness within the band, or of local bandwidth variation, so it is not the best of LU methods.
- Envelope, Block & Skyline Methods: These methods usually have a lower operations count than the Band Method since they take advantage of variations in the band, but for the same reason become difficult to load balance. The matrix for these methods is highly irregular and static load balancing schemes are not obvious. Dynamic load balancing might be used, but this would add to the communications load.

Frontal Method: This method typically has a very low operations count but is difficult to load balance (the front width continuously changes) and requires complicated bookkeeping for parallel implementation. The frontal method on a sequential machine has high bookkeeping overhead to begin with, so the additional expense of parallel logic overhead may be prohibitive.

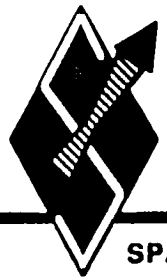


SPARTA, INC.

EXISTING SOLUTION METHODS HAVE DRAWBACKS

- ITERATIVE METHODS
  - EFFECTIVE FOR HIGH ACCURACY, 3-D ANALYSIS, ADAPTIVE ANALYSIS
  - REQUIRE MUCH COMMUNICATION
  - LARGE PROBLEMS REQUIRE FREQUENT ACCESS TO MASS STORAGE

Iterative Methods (Conjugate Gradient Method, Incomplete Cholesky Decomposition, SOR Method): Iterative methods are effective for three dimensional and higher degree element problems, problems requiring high accuracy, and adaptive mesh refinement solution methods, but typically require more interprocessor communication than direct methods. Problems too large to fit in the collective core memory of the processor network demand high throughput rates to external storage since the entire matrix must be updated at every iteration. Data transfer rates to mass storage devices could easily become the design criteria for large scale iterative solvers.



SPARTA, INC.

FIVE TRANSPUTER FE SOLVERS COSTING $200,000

	ALL 256Kb T800's	ALL 1Mb T800's	100 1M T800's + 9 DISKS	67 1M T800'S + 19 DISKS	30 1M T800's + 30 DISKS
NUMBER OF PROCESSORS	200	133	100	67	30
MFLOPS	300	200	150	100	45
NETWORK RAM	50 Mb	133 Mb	100 Mb	67 Mb	30 Mb
EXTERNAL MEMORY	---	---	1,620 Mb	3,420 Mb	5,400 Mb
LARGEST PROBLEM (D.O.F.)	8,095	13,200	47,400	67,600	84,300
TIME TO SOLVE 8,000 D.O.F.	36 sec	53 sec	71 sec	106 sec	237 sec
TIME TO SOLVE LARGEST PROBLEM	37 sec	240 sec	4.1 hrs	17.8 hrs	77 hrs

Estimates of Performance for Five Transputer FE Solvers Costing $200,000

In order to estimate the performance levels that could be obtained from a TBFES, an analysis was made of five transputer hardware configurations, each costing $200,000. The five configurations varied by (1) the amount of RAM available to each transputer and (2) the amount of external storage available to the network. Banded LUD decomposition was used as the common solution method since its execution times can be predicted with much greater accuracy that frontal or iterative methods.

The following assumptions were used to predict execution times for the five transputer FE solvers:

Notation:

N	= degrees of freedom
Xp	= number of transputers
matrix operation	= one addition and one multiplication

Prices:

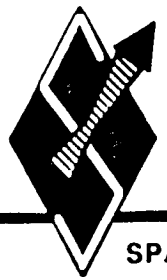
T800 with 256 Kbytes	= $1,000
T800 with 1 Mbyte	= $1,500
180 Mbyte SCSI disk drive with transputer interface	= $5,200

Assumptions:

Solution method	= Banded LU decomposition
Half bandwidth	= 0.1*N
double precision storage requirement in bytes	= 8 * 0.1N * N = 0.8N*N
T800 performance for double precision matrix operations, including overhead	= 0.6 Mflop
number of matrix operations for banded LU decomposition	= 0.1N * (N*N)/12 = (N*N*N)/120
time for LU decomposition	= (N*N*N)/(120 * 600,000 * Xp)
efficiency	= 100%

I/O transfer rates to disk do not impede performance

The estimates are based on rough calculations, but can still be used to qualitatively compare the different configurations. The solver with the most transputers, and therefore the most Mflops, appears to be the fastest of the five FE solvers, but its high granularity suggests that the 100% efficiency assumption probably will not apply. The 200 transputer network also has relatively little memory so it can only be used for problems with fewer than 8,000 degrees of freedom. The FE solver at the other end of the spectrum, the 30 transputers with a disk drive apiece, has such low granularity and such a high transfer rate to external storage that high efficiency is certain. Such a configuration would be useful for solving the largest problems but is four times slower than the 133 transputer system which can also solve problems of considerable size. The optimum combination depends entirely on user requirements; if only small problems are anticipated, or if solution speed is critical, the 133 transputer network seems ideal. A demand for the solution of very large problems would most likely call for the network with the most external storage.



SPARTA, INC.

CONCLUSION

- SBIR STUDY DEMONSTRATED FEASIBILITY OF A TFES
- PROJECTION OF RESULTS SUGGESTS A LARGE-SCALE TFES WITH SUPERCOMPUTER POWER COULD BE BUILT FOR THE PRICE OF A MINICOMPUTER
- TAPPING ALL THE POTENTIAL POWER OF A TRANSPUTER NETWORK IN A LARGE-SCALE TFES
  - APPEARS ACHIEVABLE
  - REQUIRES INTENSIVE RESEARCH AND DEVELOPMENT EFFORT
  - COULD REVOLUTIONIZE FE ANALYSIS

nclusion

The SBIR study determined that the implementation of the FE Method on a network of transputers is highly feasible. Reasonable projections suggest that a large scale FE solver with Cray-level power could be assembled for $100,000. An intensive research and development effort is required to effectively implement such a powerful FE solver, but the result could revolutionize finite element analysis by making supercomputer analysis capacities widely available to scientific, research and engineering communities.